**bigger files for xv6**

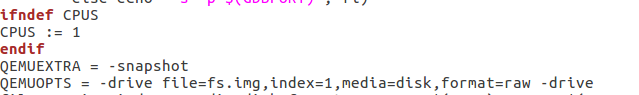
#### 一、实验目的

增加xv6文件的最大大小。当前，xv6文件限制为140个扇区或71,680字节。此限制来自以下事实：

一个inode有12个direct指针，这个指针是直接指向硬盘中的文件块的，还有一个inderict指针，这个指针指向另外一个indirect block, 这个indirect block有128的新的指针，分别指向一个文件块。这样，一个inode就可以指向：12+128=140个文件块，因此，运行big函数后会显示140个block，实验将更改xv6文件系统代码，以在每个inode中支持“双重间接”块，其中包含128个单间接块地址，每个单间接块又可以包含最多128个数据块地址。实验结束后，一个文件最多可以包含16523个扇区（约8.5兆字节）。

#### 二、实验准备

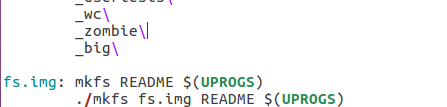
1. 修改Makefile文件：将CPUS的值设为1，并在QEMUOPTS前添加QEMUEXTRA = -snapshot语句。这样做的目的是加快xv6创建大文件时的qemu的运行速度。



2. 修改param.h文件，将文件系统的最大数据块数设置为20000。



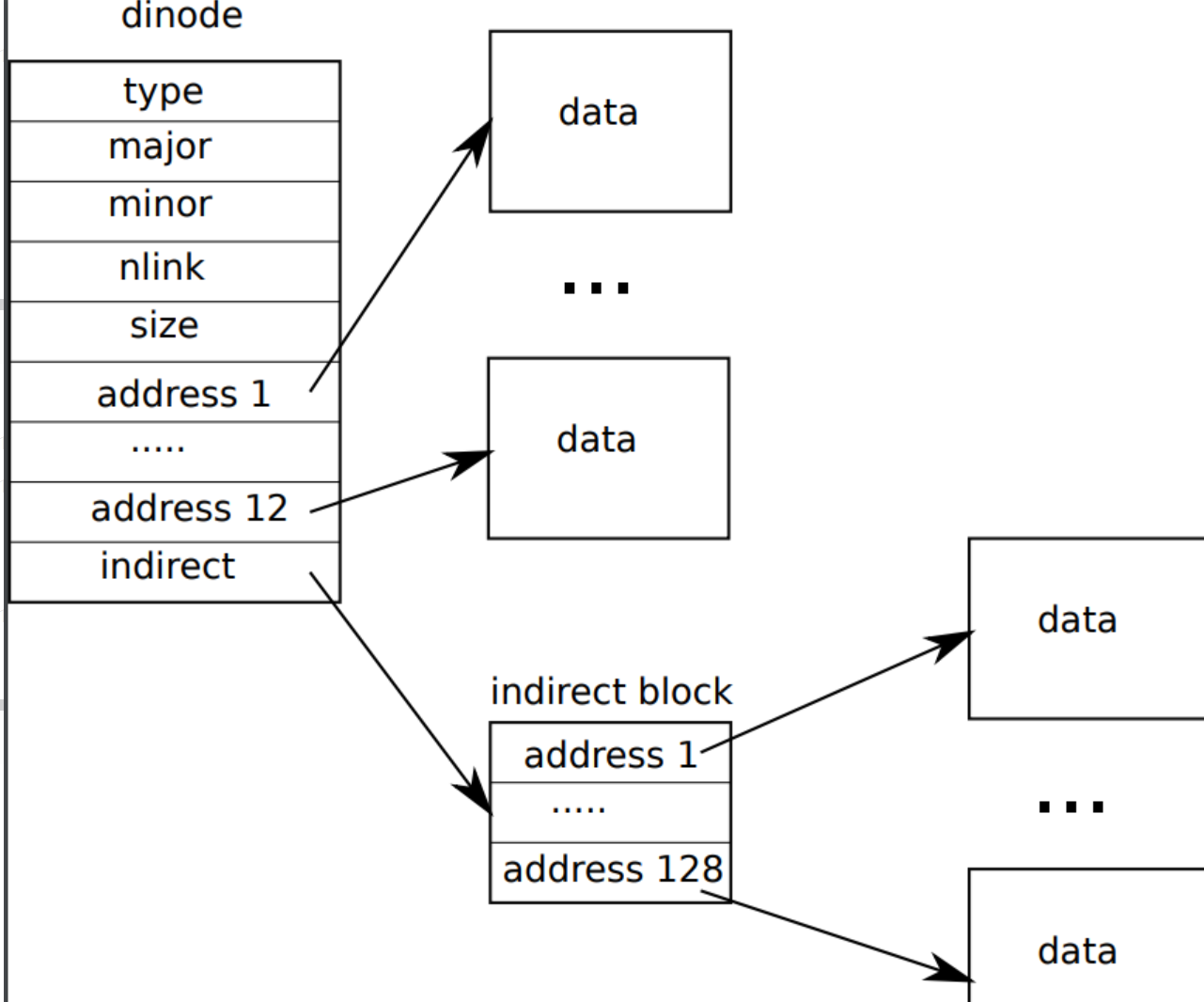
3. 下载big.c文件放入xv6的目录下，并在Makefile文件中的UPROGS列表中增加对big命令的引用：



4. 启动xv6，输入big命令，显示“wrote 140 sectors”，准备工作已完成：

#### 三、实验分析

磁盘上inode的格式由fs.h中的struct dinode定义。对NDIRECT、NINDIRECT、MAXFILE和struct dinode的addrs[]字段要感兴趣。xv6标准inode的结构图如下所示：



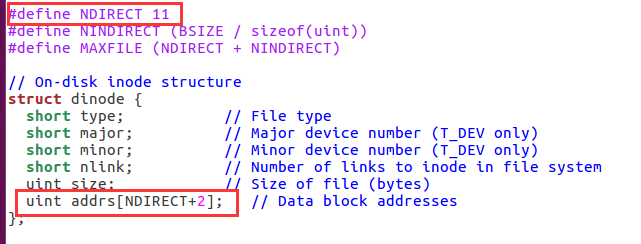
在读取和写入文件时都调用bmap()。当写文件的时候，bmap()根据保存文件内容的需要分配新块，如果需要保存块地址，则分配一个间接块。

bmap()处理两种类型的块号。bn参数是一个“逻辑块”( "logical block")——一个相对于文件开头的块号。ip->addrs[]中的块号和bread()的参数都是磁盘块号。可以将bmap()视为将文件的逻辑块号映射到磁盘块号。

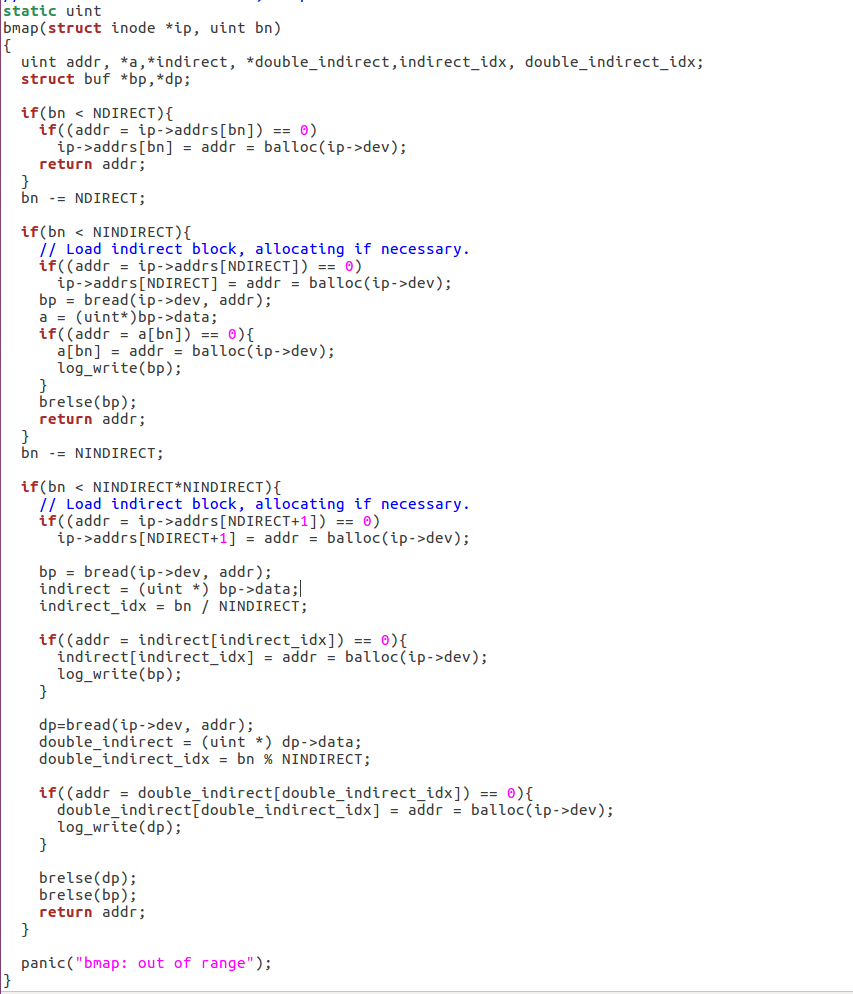
理解原理之后很容易发现，本次实验只需要把原来的12+128改成11+128+128\*128即可。

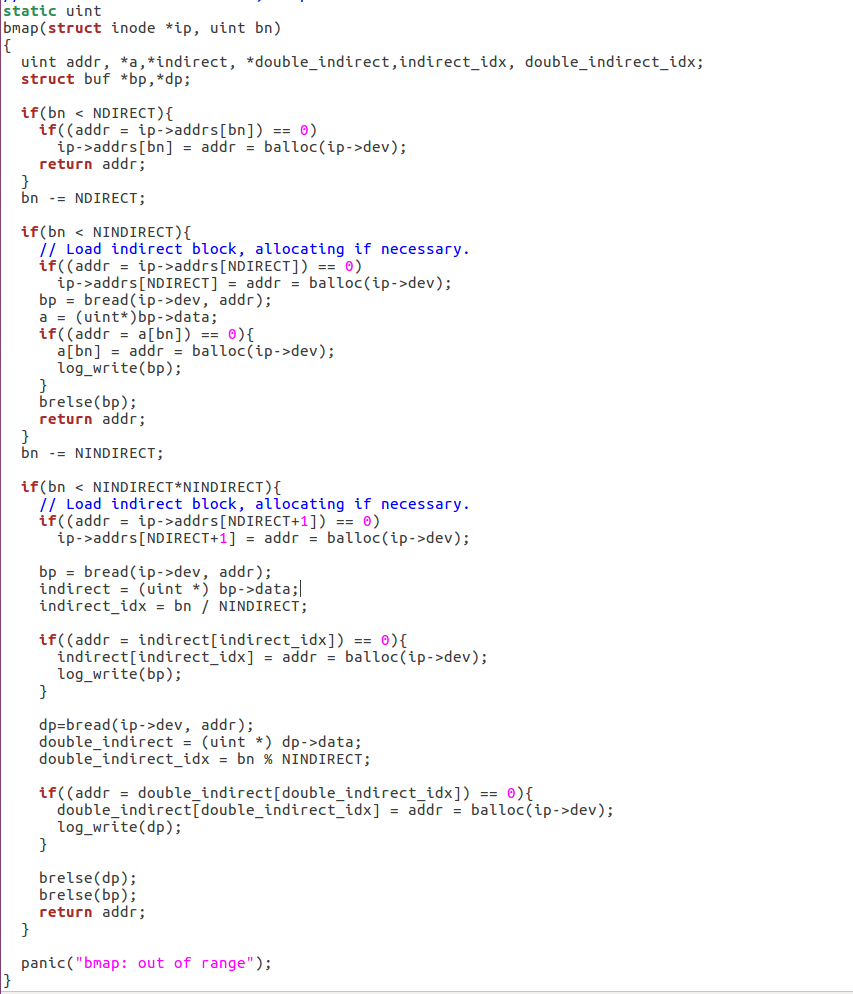
#### 四、实验过程

1. 首先修改fs.h文件。

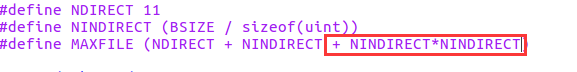
将直接块数量改为11，对应的数据块地址的索引由加1变为加2。

2. 之后，修改fs.c中的bmap()函数，在函数中增添了有关二级索引指针的内容。





3. 运行big，此时显示结果为138个block。检查出错原因，发现是fs.h文件中定义的MAXFILE太小了，在加入二级指针之后，需要把它扩大：



4. 修改之后再次运行big，得到正确的结果16523。

